UDC 004.6:004.932’328 **doi: 10.32620/reks.2025.1.10**

**Stanislav DANYLENKO, Serhii SMELYAKOV**

***Kharkiv National University of Radio Electronics, Kharkiv, Ukraine***

# DEVELOPMENT OF A MULTIDIMENSIONAL DATA MODEL FOR EFFICIENT CONTENT-BASED IMAGE RETRIEVAL IN BIG DATA STORAGE

*The **object** of the study is content-based image retrieval. The **subject** of the study is the models and methods of content-based image retrieval in Big Data storage under high-intensity search queries. The purpose of this study is to develop a multidimensional data model and related search methods that can use and adapt to existing image descriptors and perform searches based on them. The **task** is to: analyze modern approaches and solutions for effective content-based image retrieval, formulate the problem and requirements for the search system; develop a model that will effectively process descriptors and place them inside in such a way as to minimize the number of descriptors with which comparisons need to be made during the search; develop a search algorithm; develop metrics, perform experiments and compare the results obtained with analogs. The **methodology** includes analyzing the search process and highlighting the stages of descriptor formation, its placement in the model, determining the level of similarity and comparing and forming the results; building a data model and placing it in memory; conducting experiments with data sets available on the Internet; evaluating the effectiveness of the search and forming the resulting tables for comparison with analogs. The following **results** were obtained: Multi-Dimensional Cube (MDC) model with optimizations and search algorithms was developed. It was compared with the brute-force search and the search that uses Inverted Multi-Index (IMI). The experimental results showed that MDC provides the best search speed among competitors. Demonstrates search quality at the level of competitors. The search labor intensity shown by the MDC is the best for searching for original images in the storage (checking whether they are present in storage). The labor intensity of searching for modifications of the images is better than in brute-force search by more than 100 times, but worse by 30% than when using IMI. **Conclusions**: The developed MDC model with its search algorithm solves the task of efficient content-based image retrieval, using existing image descriptors. The obtained results are satisfactory, but a promising direction is to improve the cell boundaries optimization algorithm and apply parallel computing.*

***Keywords:*** *multidimensional data model; search model; content-based image retrieval; big data; image processing; image storage; feature database.*

## 1. Introduction

### 1.1. Motivation

Searching on the Internet is an everyday operation and is performed very often. According to statistics from Google, one of the leaders in web search, approximately 8.5 billion queries were performed daily in 2024, with 84% of users performing at least 3 searches per day. The image search function is used approximately 12 billion times a month. That is, image search accounts for approximately 5% of all user searches [1, 2]. Furthermore, this is information from just one search engine.

For the average user, the image-based search (when the search query is an uploaded image) may not always be in demand. Because in many cases, their query can be expressed in words. However, sometimes a user may not know what exactly they should find, having only a graphical representation of a scene or object. In this case, searching by image content is the only possible option. Popular search engines are set up to detect objects in images and search by them. As we can see from the usage statistics, they do well.

In some areas, this function can be used in professional activities, for example: in medicine to help specialists identify health problems, in face recognition to identify a person's identity, in e-commerce to search for products, or in smart home systems to compare patterns with the current state and perform appropriate programmed actions [3, 4].

In some areas, this functionality does not play a key role, and inaccurate results do not cause problems, for example, if we are talking about searching for a product or clothing on a marketplace. In such cases, the user is probably happy to get the result quickly and check its relevance on their own. They may be interested in similar products if no exact match is found. If necessary, they will make a second request, for example, using a photo of the product from a different angle or capturing product features such as a brand logo, tag, labeling, etc.

However, for example, in the medical field, the results must be of a high level of quality to reduce the number of irrelevant comparisons. As can be seen in recent research in this area, indeed, more specific tasks require specific solutions and the use of general search engines is not possible for them [4].

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Another important nuance is that public search engines, such as the aforementioned Google, scan websites available on the Internet and index all information allowed by the owners of these sites. However, regarding the professional use of search engines, an important possibility is to perform searches among corporate, often confidential, data that is closed to public access. Furthermore, since most image descriptors and search models are aimed at wide use, their use may not be effective in these specific scenarios.

There are 2 main characteristics of content-based image search: search speed and quality. When implementing such a search, we should balance between these characteristics and try to satisfy them both. The quality of the search depends directly on how the images were described. Currently, there are a large number of image descriptors for this purpose, and the main task of search models is to use them effectively [5].

Problem statement. There is a problem with efficient content-based image search using existing image descriptors for specific highly specialized areas, where search systems must adapt to ensure not only high search speed but also quality. The search should be performed in Big Data storage and be able to process high-intensity search queries. By solving this problem, we can obtain a search engine that can quickly and efficiently search for similar images in real-time among the currently available storage. Many of which can be classified as Big Data. At the same time, it should use already developed image descriptors and be able to work and tune to work efficiently with a specific data set and have a simple process of software deployment and use.

### 1.2. State of the art

The fields of content-based image search (CBIS) and content-based image retrieval (CBIR) have been studied since the last century. Long F. et al. described the development of this field from the 1990s to 2003 [6]. Zheng L. et al. and Li. X et al. in the period from 2003 to this day [7, 3].

The basic principles and stages of the search were defined in 2003 and have not changed much since then. The fundamental scheme of the search process is shown in Fig. 1.

At first, actions are performed offline an additional storage is formed to the image storage feature database (FDB). It is filled with image descriptors from the main storage. They are placed in the FDB certainly so that they can be effectively retrieved [6].

The image search is an online part of the process. During the search, the user uploads an image for which a descriptor is calculated and, using a certain similarity measure, is compared with the descriptors available in the FDB according to a certain algorithm. The found descriptors form the resulting list, sorted in order of similarity [6].

The main differences present in the existing solutions are as follows: 1) the use of different image descriptors; 2) different structure of the FDB and the algorithm for finding nearest neighbors in it; 3) different approaches to calculating the measure of similarity of descriptors.

The image search system (engine) is a software system that has a user interface for uploading an image for search and displaying the found results. It is connected to the main and auxiliary image storage. It has a layer for extracting descriptors from the image and a model for retrieving similar images from the storage by the calculated descriptor.

The search model is an abstract component of the search system that defines how it: interprets a search query; manages data: contains methods and algorithms for processing data, organizing and searching it in the data structures of FDB; and ranks results.

CBIS is the process of finding similar images based on their visual content from the perspective of a user interacting with a search engine.

CBIR is the process of retrieving similar images from a storage based on their features described by a descriptor. CBIR is usually a part of CBIS.

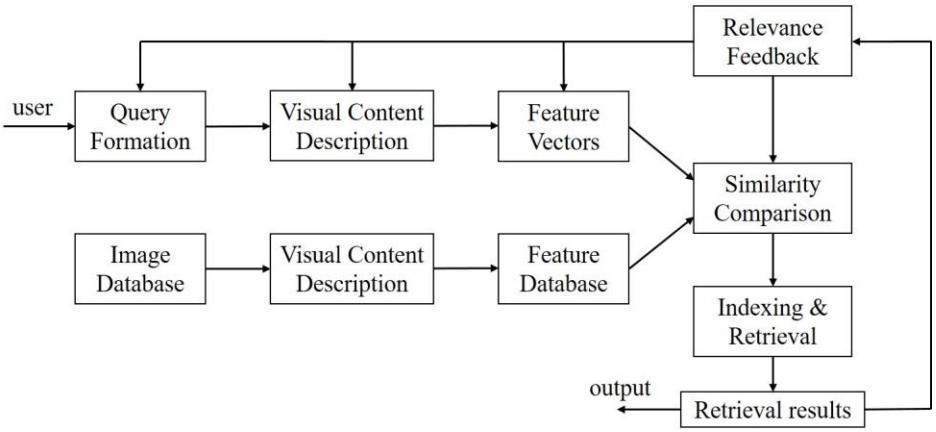


Fig. 1. Main stages of CBIR [6]

An image descriptor is a simplified description of certain features of an image presented in a certain format, such as a one-dimensional or multidimensional vector. It can be either homogeneous, when all values describe the same feature, or heterogeneous. One of the main properties of the descriptor is invariance – the descriptor does not change (or changes slightly) when the image is modified.

In the 1990s, descriptors were a verbal description of the image that was inserted into the simple DB and the search was a simple text search based on the description of the input image [6].

As the number of images grew, it became impossible to manually process all of them, so it became necessary to automatically determine the visual descriptors of the images. Since 1997, many different descriptors have appeared. They can be divided into global descriptors, i.e., those that apply to the entire image, and local descriptors, i.e., those that describe a specific point or region of the image [6]. Such descriptors are still used nowadays.

Global descriptors include the following descriptors: color (Color Space, Color Moments, Color Histogram, Color Coherence Vector, Color Correlogram, Invariant Color Features), shape (Moment Invariant, Turning Angels, Fourier Descriptor), texture (Tamura Features, Wold Features, SAR Model, Gabot Filter Features, Wavelet Transform Features) and spatial layout [6].

A descriptor can also combine information about various image features. Such descriptors, after formation, have a multidimensional form. Then, a reduction is performed to reduce the number of dimensions or to bring it to a one-dimensional vector [8].

Local descriptors are more complex to create and have been actively used since the Bag of Words approach and its analogs, such as Fisher Vector and VLAD, were applied to generate image feature vectors [9]. The existing visual words – key points or patterns – are extracted from the image, and histograms of their frequency of occurrence in the image are generated. As a result, a one-dimensional vector is obtained. The length of the vector is the number of visual words, and the value is the normalized frequency of the word in the image. Usually, it is based on the SIFT descriptor or its analogs/modifications such as SURF and ORB. This allows us to use more information from the image and make the descriptors more invariant but increases the search complexity due to the increase in the length of the vector [7, 8].

Neural networks are also often used to extract features from an image. They are also used to obtain both global and local descriptors. For example, on the basis of previously created and trained networks for object classification or detection [10, 11]. Some networks are also being created specifically for CBIR, in which the main task is to obtain similar vectors for similar images, not classification [12].

The process of retrieval of similar images by descriptors takes place depending on the form of the search model's FDB.

The following multidimensional data structures can be used to place and further search among multidimensional descriptors: R-tree, linear quad-trees, K-d-B tree, grid files and Self-Organization Map (SOM). They use descriptors in their original form, without any modifications. However, their effectiveness decreases with the increase in the number of dimensions and data volumes, which is difficult to adapt to use in modern conditions [6].

Another approach is to use the hashing of descriptors. Hashing can be performed without training, for example, using Local-Sensitive Hashing, in which vectors are converted by a hash function into values and these values must be compared during retrieval [13]. Algorithms with learning are also used to select an effective hash function based on pre-marked data [14]. However, such a search can give inaccurate results due to the properties of the hash functions used to distribute the values.

Another widely used approach is clustering. Each descriptor is assigned to a cluster or clusters according to certain rules. Now, for each cluster, there is a list of descriptors that belong to it. For an input descriptor vector, the cluster to which it belongs and among which it is to be searched is also determined. This significantly reduces the number of descriptors that need to be compared [15]. This approach, called the inverse index IFI (IVF) or inverse file, is widely used and has many variations [16].

The most modern evolution of the last described approach is the use of Product Quantization (PQ) [17] and its numerous modifications, for example IVFADC-R [18] IMI [19] or OPQ [20]. This approach allows us to form clusters for parts of the descriptor vector and represent them in a compact form. Thereby simplifying the search for descriptors in clusters and further reducing the number of descriptors to be compared. Machine learning techniques are also used for this approach.

Multidimensional descriptors can be compared using the following measures: Minkowski-Form Distance, Quadratic Form Distance, Mahalanobis Distance, Kullback-Leibler Divergence and Jeffrey-Divergence [6]. To compare one-dimensional vectors of large size, we can use the Hamming metric for binary values and the Euclidean, Manhattan distance or cosine similarity for normalized values [8, 9]. Specific comparison approaches can be used to calculate a similarity measure for particular descriptors. For example, bin matching can be used for Color Histogram based descriptors [21].

The choice of architecture and specific parameters for a CBIR system may depend on many factors, such as the specifics of the images to be processed or the available resources for the search system [22]. The descriptor

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

implementation can be chosen based on the analysis of the features of a particular set of images in a storage performed by a feature selection system. It determines the properties that allow to better determine the similarity and dissimilarity of images among themselves [23].

For search models that can be trained or adapted, Relevance Feedback is used. This technique allows users to choose which results are more relevant and thereby, for example, change the weights within the model or modify the search query to get more relevant results [5, 24].

The main metrics of CBIR systems are: precision – the ratio of the number of validly found images to all found images, and recall, the ratio of the number of validly found images to the number of all valid images in the storage [5, 24].

Currently, there are software solutions that implement some of the approaches described earlier, such as software libraries: FLANN, which can work with different search models [25], Faiss based on IMI approaches [26], and LIRE, which works based on Lucene indexing technology [27].

### 1.3. Objective and Approach

The objective of this work is to develop a search model with a special structure of the FDB and related software for efficient content-based image retrieval in Big Data storage with a high intensity of search queries. This structure has the form of a multidimensional cube. It operates on image descriptors. The model can be used with various search engines as an add-on. It should solve the problem of inefficient content-based image search.

The main idea of the presented model is to create a special FDB in the form of a multidimensional cube. Processing image descriptors and placing them inside it in such a way that similar descriptors fall into the same cell of the cube or into neighboring cells. Thus, it significantly reduces the number of descriptors with which to compare during the retrieval and simplifies the nearest neighbor search algorithm.

Such a search model should meet the following criteria/requirements:

- be universal in terms of using various descriptors, both existing and created specifically for it;
- be adaptive to the properties of a particular type of descriptor;
- be able to be customized depending on the availability of resources at the workstation;
- ensure a balance between the search speed and search quality;
- be straightforward to set up and deploy, and can be used as an add-on to existing search systems.

To accomplish this objective, the following tasks need to be performed:

- develop a general approach to performing an effective search using image descriptors;
- develop a multidimensional data model for retrieval that uses image descriptors and an algorithm for processing the descriptors;
- develop methods for optimizing and adapting the model to descriptors;
- develop methods of placing the model in the computer memory;
- develop an algorithm for retrieving and comparing descriptors in the relevant FDB.

With the presented model implemented, we can use existing image descriptors, quickly deploy the software system for various target users, and perform searches quickly and efficiently.

## 2. MDC model

### 2.1. Summary

The model that we propose is called the Multidimensional Cube (MDC). The search model follows the classical principles and stages of the search process, as shown in Fig. 1. However, it has its own peculiarities. Fig. 2 shows a general scheme of the MDC operation at the formation and search stages.

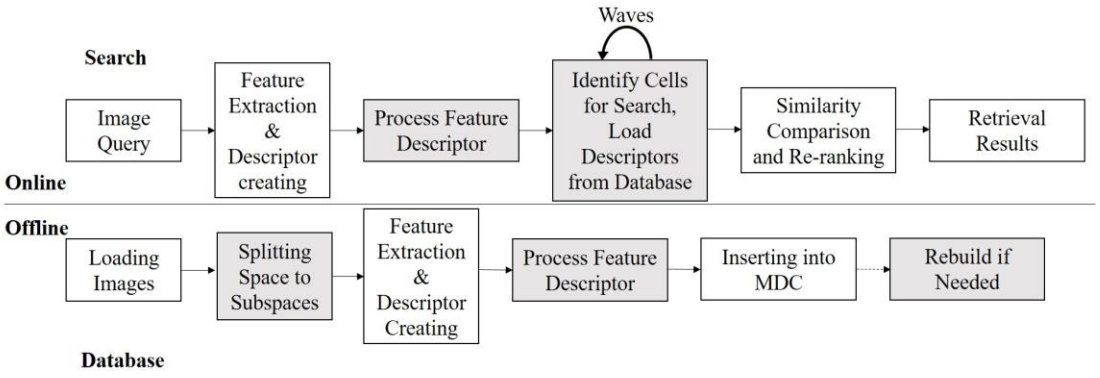


Fig. 2. General scheme of the MDC operations

There are several differences from the classical process shown in Fig. 1. For the offline part, a stage of splitting the space into subspaces and the ability to rebalance the model under certain conditions are added. For the online part, a wave-search stage is added, which will be described below. For both parts, a descriptor processing stage is added.

The Relevance Feedback stage is not supported. All other stages are present and can be named or grouped differently in the diagram compared to Fig.1.

The MDC model does not include the stage of analyzing a user's request and, accordingly, creating an image descriptor. Instead, it uses ready-made descriptors or libraries to generate them. This is done to make the model adaptive and able to be widely used with different search engines and descriptors. The required descriptor handler can be added as a filter in the search engine. As a result, any descriptor that has the format of a one-dimensional vector can be used for MDC.

The principle of MDC operation is described in Section 2.2. Methods of MDC optimization are described in Section 2.3. Methods for placing MDCs in memory are given in Section 2.4. The algorithms for comparing descriptors and searching inside the MDC are given in Section 2.5. Comparisons with existing search models are given in Section 2.6. The software implementation is described in Section 2.7. The methodology for comparing the models using experiments is given in Section 3.

## 2.2. MDC Structure

An image descriptor is a one-dimensional vector of a certain length. From a geometric point of view, the values of this vector define a point in a multidimensional Euclidean space. That is, each value corresponds to a coordinate in a certain dimension. If several descriptors have the same vectors, they are located at the same point and form a cluster of identical images. The values of the vectors are usually normalized, i.e., they are within a certain range, for example, [0-1], and are fractional numbers, i.e., they are not discrete. Thus, in the initial form, we have a large (tending to infinity) number of unrelated clusters, where each cluster contains a small number of descriptors.

MDC solves this problem and allows us to form clusters of descriptors efficiently and in such a way that there is a relationship between them. This is achieved through the previously mentioned stages of space partitioning into subspaces and descriptor processing. This is necessary for the functioning of the multi-dimensional FDB, which is the basis of MDC.

The process of dividing a space into subspaces consists of two stages. The first stage is to reduce the number of dimensions in which the MDC is located. Their number corresponds to the length of the descriptors. Descriptor vectors usually have a large length, but for efficient placement and retrieval, this length should not be large. In MDC, the length of the vector, and hence the number of dimensions, is denoted as $N$ and is determined at the configuration stage. This completes the first stage of partitioning.

The second stage is to divide the scale of the possible descriptor values in each dimension into intervals. We use these intervals to form an additional descriptor vector a vector of indices of the interval by dimension.

The set of possible coordinate values becomes finite and clearly defined. The number of intervals is also determined at the configuration stage and is denoted as $k$. We obtain $N$ dimensions, each of which is divided into $k$ intervals, thereby dividing the initial space into clearly defined subspaces.

Initially, the intervals are formed uniformly within the possible normalized values. If the range of the possible values is [0-1], then the intervals will be as follows: [0-0.25), [0.25-0.5), [0.5-0.75), [0.75-1]. Then, an optimization is performed that forms the boundaries individually for each dimension. So that each interval contains the same number of descriptor values that are currently in the MDC. The parameters $N$ and $k$ determine the internal structure of the MDC. The principle of their selection and the algorithm for optimizing the boundaries of the intervals are discussed in the next subsection.

From a geometric point of view, the MDC bound the space by dividing it into cells. The cells represent hypercubes, and after optimization, hyperparallelepipeds. Each of these is a cluster of descriptors. From a practical point of view, it determines the multidimensional data structure of the FDB, in which descriptors are placed in an easy-to-search form

The total number of MDC cells is determined by the following formula:

$$cc = k^N, \quad (1)$$

where cc – number of cells;

N – number of dimensions;

k – number of intervals.

The structure of the MDC is defined. To use descriptors, they must first be processed based on the defined parameters $N$ and $k$. Vectors usually have a length equal to the power of two, so the desired length $N$ can be achieved by the pairwise aggregation of adjacent values. At this stage, some data loss occurs, but it will be noticeable only at the stage of placing the descriptor in the MDC. Since their original vectors are stored and used to compare the descriptors. After aggregation, the upper limit of the possible normalized values changes depending on the number of aggregation steps performed. For example, if initially, the maximum normalized value was 1.0, then after one aggregation operation it will be 2.0.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Next, for each vector value, the index of the corresponding dimension interval is determined. An additional image vector, the vector of indices, is formed from the determined index values. The coordinates of the point in space (the cluster of descriptors) are determined by natural numbers. This cluster is the MDC cell. Neighboring cell indices by dimensions mean that these cells include descriptors with similar values, which allows us to efficiently search for the nearest neighbors. This will be described more in subsection 2.5.

Let's look at a specific example. Suppose the original vector had a length of 32. After aggregation, the length was reduced to 4 ($N$ = 4) and it had the form [0.23, 0.45, 0.71, 0.01]. The dimensions in the range of possible values after aggregation [0-1] were evenly divided into 4 intervals ($k$ = 4). Then the vector of indices looks like in Fig. 3 – [1, 2, 3, 1].

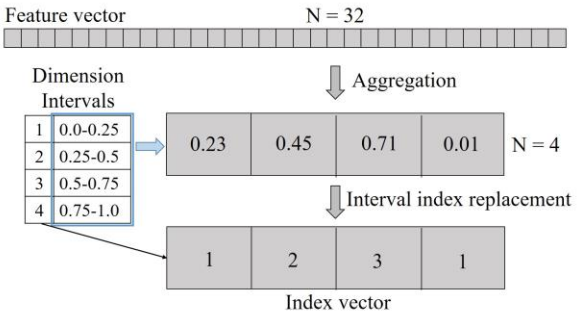


Fig. 3. How descriptors are processed

Therefore, the descriptor of this image is placed in the MDC cell with the indices [1, 2, 3, 1], where similar images will also be placed. Such a vector cannot be represented in the Cartesian coordinate system. However, if we are talking about a conditional vector of length 3 and three equal dimension intervals, such as [3, 1, 2] on the [x, y, z] axes, it can be depicted as in Fig. 4 or the MDC itself in as a geometric cube divided by cells as in Fig. 5.

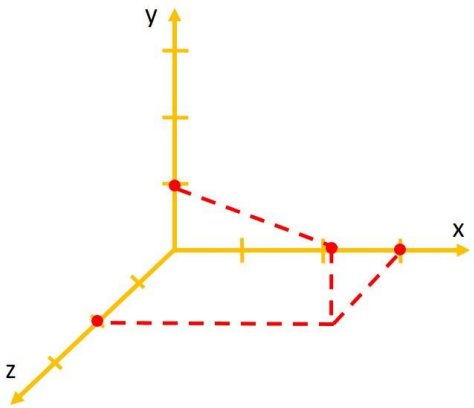


Fig. 4. MDC cell in three dimensions

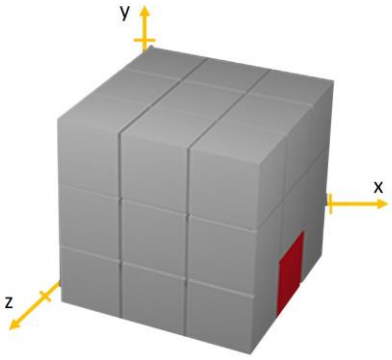


Fig. 5. MDC in the form of a geometric cube

## 2.3. MDC optimizations

In the process of building an MDC, 2 main optimizations are performed: 1) optimization of the number of cells; 2) optimization of the cell sizes (interval boundaries).

The first is necessary to efficiently use the available space on the workstation. If we create too few cells, each cell will contain a large number of descriptors. Therefore, increasing the number of comparisons during the search. If we create too many cells, they will be poorly filled or empty, which will require a large number of cells to be viewed during the retrieval.

This optimization is achieved by setting the parameters of the number of dimensions $N$ and the number of intervals in the dimensions $k$. After all, they determine the division of space into subspace MDC cells. These parameters are determined based on the number of images in the storage. They also depend on whether the system is configured to search for the original images or the original images and their modifications. If we want to always return a certain number of images (a search page) as a search result, it is advisable to set the number of descriptors in a cell equal to the size of this page. This configuration option is the most recommended.

The general algorithm for determining the parameters is as follows: 1) obtain information about the number of images in the storage; 2) determine the number of descriptors in one cell based on the search criteria; 3) select the parameters $N$ and $k$ such that:

$$rc = \frac{dc}{cc}, rc \approx ec, \quad (2)$$

where rc – number of descriptors in one cell with the current parameters;

dc – total number of image descriptors;

cc – the number of MDC cells from Eq. (1);

ec – the expected number of descriptors in one cell.

As already mentioned, $N$ is a power of two, and $k$ can be any. All possible combinations of $N$ and $k$ are tried, and the one that makes $rc$ the closest to $ec$ is chosen.

The second optimization is the optimization of the cell size. As discussed in subsection 2.2, initially, the dimension scales are divided into intervals evenly. The descriptor values are not necessarily evenly distributed between the upper and lower bounds. We perform aggregation, which can break an even distribution. Therefore, a situation arises when one part of the MDC can be filled better than another one. To solve this, we need to redistribute the boundaries of the intervals so that each interval has approximately the same number of descriptors. This is done on the basis of the descriptors added to the MDC: using all of them or a certain part of them. This operation is performed separately for each dimension.

The optimization algorithm is as follows: 1) compare the current number of descriptors within the interval with the $ec$ value; 2) if the value is less/bigger, move the upper boundary of the interval up/down with a certain step; 3) stop when the value approaches near $ec$. An example of the four redistributed intervals for the four dimensions is shown in Fig. 6.

| | Interval 1 | Interval 2 | Interval 3 | Interval 4 |
|---|---|---|---|---|
| Dimension 1 | 0.0-0.1 | 0.1-0.15 | 0.15-0.63 | 0.63-1.0 |
| Dimension 2 | 0.0-0.38 | 0.38-0.52 | 0.52-0.77 | 0.77-1.0 |
| Dimension 3 | 0.0-0.11 | 0.11-0.24 | 0.24-0.55 | 0.55-1.0 |
| Dimension 4 | 0.0-0.22 | 0.22-0.36 | 0.36-0.89 | 0.89-1.0 |

Fig. 6. Intervals after optimization

As a result, we obtain a more even distribution of descriptors across the MDC, which will ensure a predictable, almost constant search speed.

After adding a large number of descriptors, the distribution of descriptors may change and we will need to perform the optimization steps again. To do this, we do not need to modify the descriptors themselves, but to update the cell parameters and indices of the cell to which the descriptor belongs.

The cell boundary optimization algorithm and its modifications will be discussed in more detail in a separate paper.

## 2.4 Placing MDC in memory

MDC is a multidimensional model because it contains a multidimensional FDB. There are different ways to fit it into a computer's memory, which is one-dimensional. Let's consider 2 possible options.

The first option is based on the already known OLAP cubes and the "star" layout scheme [28]. This approach is used to place MDC in a relational database. For each dimension, a separate table "feature_N" is allocated, in which the available intervals are stored. Descriptors are written to a separate table "descriptors" in an aggregated form along with the initial form. An additional linking table "links" is created, in which the descriptor ID and interval indices to which the descriptor values on each dimension belong are stored. The structure looks like the one shown in Fig. 7, for $N = 3$, $k = 3$.

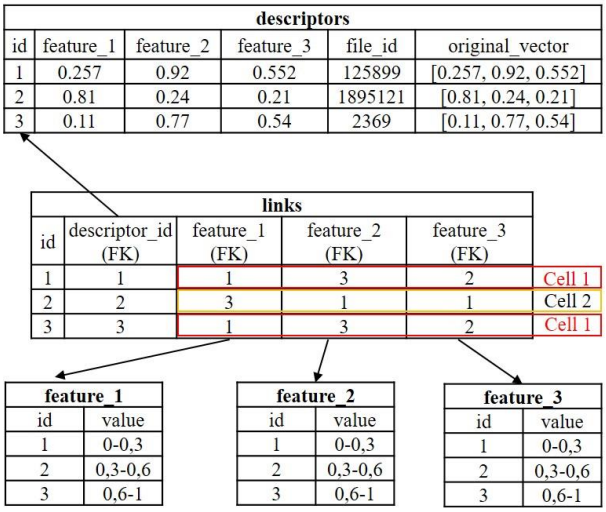


Fig. 7. Placing MDC in DB

Records with the same feature values in the "links" table indicate that the descriptors are in the same cell. Accordingly, the descriptors are retrieved by these indices and by the foreign key with the "descriptors" table.

Another approach is to place the MDC in the RAM. When placed in RAM, the MDC cells are defined as the Cartesian product of all possible MDC dimension interval indices. The cell is defined by the corresponding value of the vector of indices, written in string form. For example, the calculated vector of indices [1, 2, 4, 1] will be represented as "1-2-4-1". Therefore, in this case, MDC is represented in one-dimensional form. This implementation is based on a hash table, in which the key is the above string representation of the vector, and the value is the list of descriptors that fall into this cell. The dimension intervals are stored as a configuration separately from the MDC itself. The visualization is shown in Fig. 8.

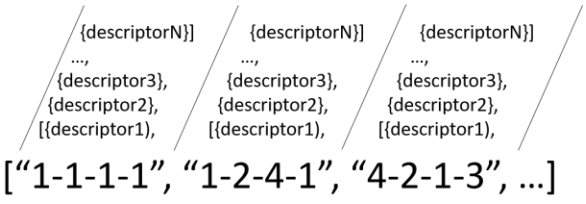


Fig. 8. Placing MDC in RAM

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

This implementation has a significant advantage in terms of model creation and optimization time, and search time. This will be discussed in more detail in Section 4. However, it requires additional storage where the model will store its state and dimensions configuration between runs. It can be a regular file or files in a shared file system, key-value storage, or cloud solutions.

The DB approach is useful if we can allocate a lot of resources to the DB, but have limited RAM. Otherwise, it is better to use a RAM-based implementation.

### 2.5. Search in MDC

MDC uses a hybrid wave-search method. The idea is to use a special order for checking MDC cells and applying a brute-force search as part of the cell inspection. The algorithm is as follows: 1) identify the MDC cell by the vector of indices of the searched image; 2) perform a brute-force search in this cell; 3) if the search was not successful in this cell, perform one search wave; 4) if the desired result was not found in the previous wave, continue the search until the maximum available number of waves is reached or until all cells are inspected.

The search wave is performed as follows. The vector of indices consists of integers. Cells that have values in their vector of indices that are closest to those specified in the vector of indices of the initially found cell are closest to it. During the search, the values of the vector of indices should be decreased and increased by 1, if possible. Furthermore, the cells with the calculated vector of indices should be checked. In this way, the MDC will check the nearest neighbors of the initially defined cell and obtain image descriptors that are as similar as possible to the one being searched for. If the search needs to be continued, another wave of search is performed, and the index value will differ from the initial one by 2 and so on.

The method is well illustrated graphically and is shown in Fig. 9 for a search in MDC with $N = 2$, $k = 10$.

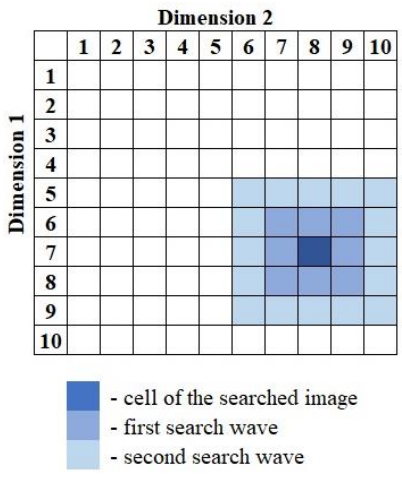


Fig. 9. Wave-search

The dimension intervals are divided evenly within [0-1] with a step of 0.1. Let the descriptor vector be [0.65, 0.73]. Then the initially found cell is [7, 8]. In the first search wave, the cell indices for dimension 1 will be (6, 7, 8), and for dimension 2 – (7, 8, 9). In the second wave, they increase/decrease by 1 more relative to the index of the initially found cell.

The cell contains a certain number of descriptors *rc* from formula (2). Then, the number of descriptors in the search wave is determined by the following formula:

$$wc_i = (j_i)^N \times rc - (j_{i-1})^N \times rc, \quad (3)$$

where wc – number of descriptors in the search wave;

i – wave number starting from 1;

j – an increasing sequence of positive odd numbers: [3, 5, ...];

N – number of dimensions;

rc – from the Eq. (2).

The similarity of the descriptors is checked using the Manhattan distance because it satisfies the requirements for universality, set for MDC:

$$\mu = \sum_{i=1}^{N} \left| v_i^{(1)} - v_i^{(2)} \right|, \quad (4)$$

where $\mu$ – the difference between descriptors;

i – ordinal number of the vector elements;

N – number of elements of the vector;

$v^{(1)}$, $v^{(2)}$ – vectors of the first and second descriptor;

The smaller the value of $\mu$ the more similar the images are to each other.

The search result is a list of identifiers of the found descriptors sorted by value $\mu$. The search algorithm and its modifications will be discussed in more detail in a separate paper.

### 2.6. Search models comparison

The simplest classical approach is the brute-force search (abbreviated as BF), where the search requires checking all available descriptors or stopping after finding specific ones if a stopping condition is specified. Compared to it, using MDC allows us to significantly reduce the number of descriptors with which to perform a comparison. It has a positive effect on the search speed. For example, if there are 100 000 descriptors in the storage and 10 000 cells have been created in MDC, then in the best case scenario, only 1 cell needs to be inspected during the search, that is, only 10 descriptors instead of 100 000. However, because not all descriptors are reviewed, the search quality may be somewhat lower.

The closest in nature to the solution we have presented is the approach using Product Quantization (PQ) and specifically the Inverted Multi-Index (IMI) model

presented in [19]. It also divides the space into subspaces. The vector is divided into subvectors. In each subspace, the k-means clustering is performed and the value of the subvector for each subspace is replaced by the index of the nearest centroid, thus also forming a vector of indices. For simplicity, we assume that this vector of indices also forms a "cell".

However, there are several important differences between the approaches: 1) clustering methods are used to determine clusters of descriptors, which is more complex than the approach we propose with the division of dimensions into intervals and optimization, 2) during the search for the input vector, the distance to all IMI cells must be calculated and then inspected in order of distance. In MDC, the order of inspection of other cells is determined by its structure and no additional steps are required. That is, in terms of time spent, the MDC approach should be more efficient.

The MDC is experimentally compared with the search models that implement these mentioned approaches.

### 2.7. Software implementation

This is just an example of what a search system that uses MDC can look like. We used this software for the experiments.

For the alternative approaches mentioned above IMI and BF search models with placement in RAM were implemented. Therefore, for the MDC, we also implemented the RAM placement to ensure the correctness of the results. For all models, the Manhattan distance given in Eq. (4) is used to determine the similarity of the descriptors.

The implementation is a software system consisting of a backend and frontend. The image descriptors are provided in a ready-made form in a csv file and loaded into the system at the configuration stage.

The backend is implemented using the Java 17 programming language, Spring Framework 3, PostgreSQL 15.2 is used as a DBMS, and a file in the kryo format is used as a storage for the MDC implementation in RAM [29]. The backend contains model implementations for all of the mentioned search approaches.

The frontend is implemented using HTML5, CSS3, Bootstrap 5, and jQuery technologies. The frontend contains a user interface for both classic image search, where a user can select an image (Fig. 10) to search for and get results (Fig. 11), and a special interface that is used for experiments. It allows us to obtain the results in an Excel file. The frontend can be used with any existing search model on the backend, but it must be specified on the initial screen.

The software implementation, without filling in the search models with descriptors, requires only 21.1 MB of RAM, i.e., the cells do not require memory until they have no descriptors. It is very easy to deploy and configure on the target workstation and requires no programming skills.

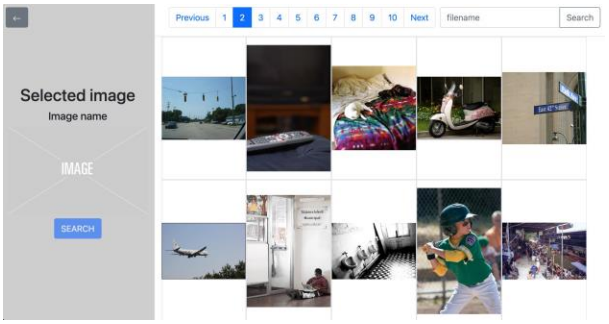


Fig. 10. Select an image to search for

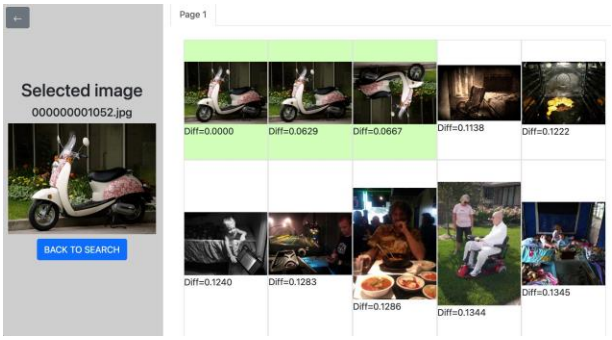


Fig. 11. Displaying the search results

## 3. Experiments Methodology

The experiments compare the approaches using the MDC, IMI, and BF models with the corresponding search algorithms. The comparison is made using the data described in Section 3.1 and the metrics described in Section 3.2. The experiments test the speed and efficiency of the model formation and search.

All experiments were conducted on a MacBook Pro 2021: M1 Pro processor on ARM architecture, 10-cores up to 3.2 GHz, 16 GB of LPDDR5 SDRAM up to 200 Gb/s, 512 GB SSD, integrated GPU with 16 cores.

### 3.1. Experiment Data

The COCO2017 dataset was used in the experiment [30]. It contains 123 403 completely different images. We randomly selected 100 000 images from the dataset. For each image, 2 modifications were created: a 180-degree rotation and a 2-fold reduction in scale. These images and their corresponding descriptors were also added to the experimental data.

To effectively describe such modifications, a special homogeneous grayscale image descriptor is used. It is a normalized one-dimensional vector of fractional

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

numbers. The descriptor contains a histogram of the brightness time-total of the image pixels. It is described by the following formula:

$$H = \{h_i\}_{i=0,\dots,2^n-1}, h_i = \frac{m_i}{m}, \sum h_i = 1, \quad (5)$$

where H – a histogram;

$h_i$ – brightness frequency of image pixels;

m – number of pixels in the image;

$m_i$ – the number of pixels of the image with brightness in the range of values with the number i;

n – integer.

The process of creating the descriptor is complex and is described in detail in a separate paper. It involves many transformations, which result in the brightness histogram taking on an invariant form, which makes the image resistant to transformations and represents the abstract image properties. For example, here is an image (Fig. 12), its brightness histogram (Fig. 13), and the descriptor histogram (Fig. 14), which is written as a vector with a length of 32.

Fig. 12. An image from the COCO dataset

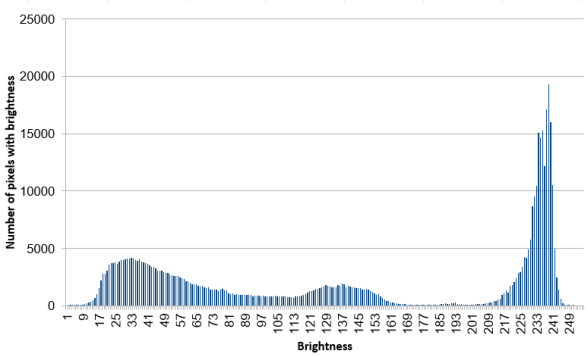


Fig. 13. Image brightness histogram

The descriptor was obtained as a one-dimensional vector of length 32, where each value is normalized in the range [0-1] and represents a 4-byte fractional number. Compared to the original dataset, which requires 19 GB for storage, the set of descriptors requires 15 MB, i.e., almost 1300 times less.

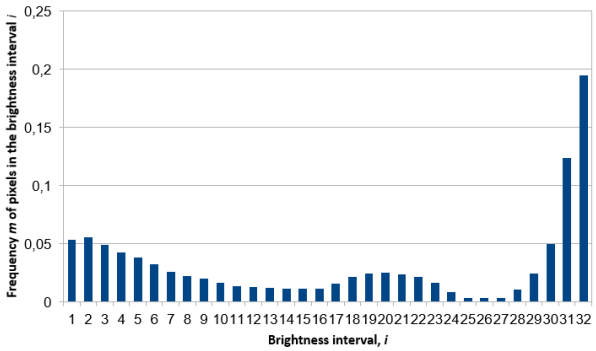


Fig. 14. Image descriptor histogram

### 3.2. Metrics

The retrieval performed in the experiments is somewhat atypical for CBIR systems. We have 100 selected images from the dataset and 2 modifications for each. The retrieval is performed until the originals and modifications are found. The first option is to retrieve only the originals, and the second option is to retrieve both the originals and the modifications. This approach to conducting experiments is designed to identify all the strengths and weaknesses of MDC, not just to analyze the results. Therefore, the use of classical metrics such as recall and precision is inappropriate, since the retrieval will not be applied in certain portions, such as 10, 100 or 1000 first-found descriptors. Other metrics are proposed.

Labor intensity (c) is the number of comparisons of descriptors with the searched one that was performed as part of the retrieval until all the searched images were found. This is a key metric under the following experimental conditions.

Search time (t) is the time for which the search was performed.

Search quality (q) is the percentage of correctly found images in the sorted list of all found images L, which also includes images that do not belong to the group of searched images and are located between the searched images. It is calculated by the formula:

$$q = 1 - \frac{p2}{p1 + p2}, \quad (6)$$

where q – search quality;

p1 – number of correctly found images;

p2 – the number of images that are between the images in the searched group and do not belong to it.

When p2 = 0 the value is 1, that is, in the resulting list, the correctly found descriptors go from the first number sequentially.

### 3.3. Experiments plan

For the selected 100 000 images, descriptors of the format specified in subsection 3.1 are created in advance. Descriptors were also created for 200 modified images. The total number of images in the storage is 100 200. The descriptors are saved in csv format and provided for the experiments.

As noted, the system has implementations of three search models that are used in the experiments: MDC, IMI, and BF.

The data analysis stage was performed to determine the MDC parameters and the number of IMI subspaces and clusters.

All descriptors are added to the system. The MDC and IMI implementations perform the stage of creating the internal structure of the FDB based on the results obtained in the previous step. For MDC, this means creating cells and optimizing their boundaries. For IMI, clusters of descriptors are created. The speed and the efficiency of the model creation are evaluated.

A search is performed in all selected models in two modes:

1) search for originals (check if the image is in the storage);

2) search for originals and image modifications. Results are presented based on the described metrics. The results are compared.

Conclusions are drawn based on the objectives and purpose of the work. The next steps for future research on the topic are set.

## 4. Results and Discussion

### 4.1. Choosing Parameters

There are 100 200 descriptors in the storage, and we need to search for the original image, or the original and two of its modifications – a group of three images. The size of the search page for the user is set to 10 images (Fig. 11). Let's set the parameter rc from Eq. (2) to 10. That is, there should be 10 descriptors in one cell. This should provide a balance between the number of descriptors in the cell and the number of cells that need to be inspected during the search.

Were found 2 combinations of parameters N and k that make the parameters rc and ec from Eq. (2) the most approximate. They are shown in Table 1.

However, with the parameters N = 8, k = 3 it is possible to perform only 1 wave of search, and we need to go through all the descriptors in the MDC. Which is unacceptable. Then the only valid configuration is N = 4, k = 10, which makes Eq. (2) as close as possible.

Table 1

Comparison of the valid MDC parameters

| | N = 4, k = 10 | N = 8, k = 3 |
|---|---|---|
| Number of cells | 10 000 | 6 561 |
| Number of descriptors in one cell | 10.2 | 15.272 |
| Number of cells in 1 wave + initially found cell | 81 | 6 561 |
| Number of descriptors on 1 wave + in initially found cell | 826.2 | 100 200 |

The same parameters are chosen for the IMI implementation. The vector is divided into 4 parts, forming 4 subspaces. In each subspace, 10 centroids are created. This creates a similar partitioning of the space as in MDC, but differently. In IMI, we will conventionally call the combinations of subspace indices as cell indices.

### 4.2. Optimizations

At the stage of building MDC, cell boundaries are optimized, and for IMI, clustering within subspaces is performed using the k-means method.

Table 2 demonstrates the evaluation of the distribution of descriptors across cells – the number of cells with a certain number of descriptors.

Table 2

Distribution of descriptors by cells

| Number of descriptors in a cell | The number of such cells in the MDC | The number of such cells in IMI |
|---|---|---|
| 0 | 7 232 | 7 417 |
| 1-10 | 987 | 1 506 |
| 11-20 | 444 | 315 |
| 21-30 | 277 | 154 |
| 31-40 | 212 | 116 |
| 41-50 | 170 | 91 |
| 51-100 | 446 | 188 |
| 101-200 | 202 | 94 |
| 201-300 | 25 | 50 |
| 301-400 | 3 | 25 |
| 401-500 | 1 | 14 |
| 501-1000 | 1 | 21 |
| 1001-2000 | 0 | 9 |

Table 3 shows a comparison of the optimization results: optimization speed, the number of filled cells out of 10 000 created, and the maximum number of descriptors in one cell.

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

Table 3

Comparison of the optimization results

| | Time, sec | Filled cell count | Max descriptor count in cell |
|---|---|---|---|
| MDC | 1.72 | 2 768 | 723 |
| IMI | 43.56 | 2 583 | 1 788 |

### 4.3. Search

After the optimizations, the search is performed in MDC, IMI, and BF. It consisted of 2 stages for 100 previously described images:

1) searching for the original images;

2) searching for the original and two modified images of the original. Each experiment was conducted in 10 rounds.

Between the rounds, the descriptors in the cells were shuffled, thereby emulating a real-world scenario where the searched descriptors can be located in any position within the cell. The results in the tables are presented in order: minimum, maximal, and average data of the metrics described in Section 3.2, and using hardware described in Section 3. Each result represents the average value of the search metrics for 100 images in 10 rounds. The results are shown in Table 4 for the searching of originals and in Table 5 – for the modifications.

Table 4

Results for searching the originals

| | Labor intensity (c), count | Quality (q), [0-1] | Search time (t), sec |
|---|---|---|---|
| MDC (min, max, av) | 1<br>373.4<br>47.82 | 1<br>1<br>1 | 0.000007<br>0.000396<br>0.000039 |
| IMI (min, max, av) | 1.1<br>1 275.5<br>166.816 | 1<br>1<br>1 | 0.004148<br>0.006177<br>0.004360 |
| BF (min, max, av) | 940.1<br>99148.3<br>49 717.161 | 1<br>1<br>1 | 0.000505<br>0.081146<br>0.037139 |

Table 5

Results for searching the originals and modifications

| | Labor intensity (c), count | Quality (q), [0-1] | Search time (t), sec |
|---|---|---|---|
| MDC | 3.3<br>3 125.4<br>644.158 | 0.089<br>1<br>0.973 | 0.000006<br>0.002993<br>0.000470 |
| IMI | 3<br>6 693.2<br>453.603 | 0.084<br>1<br>0.965 | 0.004153<br>0.016679<br>0.004601 |
| BF | 20 476.8<br>99 823.2<br>75 554.386 | 0.132<br>1<br>0.966 | 0.014517<br>0.085602<br>0.057339 |

When searching for originals in MDC, only the cell determined by the descriptor vector of indices was always checked. That is, the number of descriptors to be compared is determined by Eq. (2). For the search for originals and modifications based on the results obtained in MDC, no more than 1 search wave was always made. That is, the number of descriptors determined by Eq. (3) for i = 1.

Thus, theoretically, the expected labor intensity in MDC for searching for originals is 10.2, and for searching for modifications – 816. These results demonstrate the maximum possible labor intensity.

In addition, an MDC with an ideal distribution of descriptors was constructed and artificially inflated. In this model, by analogy with Table 3, the number of cells with the number of descriptors 1-10 is 7 322, and with the number of descriptors 11-20 is 2 678. The maximum number of descriptors in one cell is 18. No optimization is performed because the descriptors, except for the selected 100 and their 200 modifications, are specially created with values to fit into a specific cell. All 10 000 cells are filled. This is the ideal filling for MDC, when its performance is the highest and its labor intensity is the lowest, as opposed to the theoretical estimates. The search results under these conditions are shown in Table 6.

Table 6

Experimental results in an artificially filled MDC

| | Labor intensity (c), count | Quality (q), [0-1] | Search time (t), sec |
|---|---|---|---|
| Search for original images | 1.2<br>14<br>6.93 | 1<br>1<br>1 | 0.000007<br>0.000217<br>0.000016 |
| Search for original and modified images | 6<br>775.8<br>156.167 | 1<br>1<br>1 | 0.000006<br>0.001498<br>0.000165 |

These results are purely theoretical and almost impossible to achieve in reality since it is very difficult to create a descriptor that will have good invariant properties for retrieving image modifications and at the same time ensure a uniform distribution of values.

### 4.4. Discussions

In this study, we developed an approach to content-based image retrieval using image descriptors. The proposed approach differs from classical implementations by specially dividing the descriptor feature space into subspaces, thereby creating a multidimensional data model for retrieving; special processing of descriptors for use in this model and a special wave-search algorithm and the ability to rebuild the model as needed.

The main feature of MDC is its ability to work with a large number of different types of descriptors, process them, and adapt to the specifics of a particular descriptor and the number of descriptors in the storage.

The number of MDC cells is determined by a simple algorithm, depending on the preferences of the search system administrator and the intended use of the search system. The boundaries of MDC cells were optimized based on the descriptors available in the storage.

MDC differs from similar approaches in the speed of creating or updating the feature database: 1.72 s vs. 43.56 s compared to IMI, a difference of 25 times, which is a significant indicator for large amounts of data.

The efficiency of filling cells was also slightly better: 2 768 (MDC) filled cells vs. 2 583 (IMI). This demonstrates that on average, fewer descriptors are placed in one cell, 36.1 (MDC) vs. 38.7 (IMI). The maximum number of descriptors in a cell is also positively different: 723 (MDC) vs. 1 788 (IMI). Only about 30% of MDC or IMI cells are filled in. Since the value is similar for both approaches, it can be argued that the descriptors are distributed in this way precisely because of their specificity. In IMI, there are more cells containing 1-10 descriptors, but unlike MDC, there are significantly more cells with more than 300 descriptors and cells with more than 1 000 descriptors.

MDC requires few resources to deploy and has several options for memory placement. With this approach, MDC takes up 1 300 times less memory than placing the original images and searching them directly. The software implementation itself takes up another 21.1 MB of RAM, which slightly degrades the above advantage number.

A special wave-search method has been developed based on the use of processed descriptors that are placed in the MDC cell by dimension interval indexes. Experiments have shown its high efficiency. The search quality in all the considered approaches is at the same level: for the search for originals, it is always equal to 1. For the search for originals and modifications, the differences between the approaches are at the level of error.

A comparison of the gain in speed for the search for originals is shown in Table 7, and for the search for originals and modifications is shown in Table 8. The tables include the results of the search using an artificially filled MDC (called MDC Synthetic).

Table 7

Comparison of the gain (MDC, IMI vs. IMI, BF) in the time of searching for originals, in times

| | IMI | BF |
|---|---|---|
| MDC | 111.795 | 952.282 |
| MDC Synthetic | 272.500 | 2321.188 |
| IMI | - | 8.518 |

Table 8

Comparison of the gain (MDC, IMI vs. IMI, BF) in the time of searching for modifications, in times

| | IMI | BF |
|---|---|---|
| MDC | 9.789 | 121.998 |
| MDC Synthetic | 27.885 | 347.509 |
| IMI | - | 12.462 |

MDC is significantly faster than IMI and BF when searching for originals – by 111.795 and 952.282 times, respectively, and faster for searching for modifications by 9.789 and 121.998 times, respectively. The experiment with an artificially filled MDC showed that in the case of an absolutely uniform distribution of descriptors within the MDC, it is possible to achieve a gain of 3 times the results obtained with both IMI and BF.

The results of the gain in terms of labor intensity are shown in Table 9 for the search for originals and in Table 10 for the search for modifications. The tables also include the results of performing a search using an artificially populated MDC (MDC Synthetic) and a theoretical evaluation based on the results, which suggest that 1 search wave is sufficient to find all available image modifications (MDC Theoretical).

Table 9

Comparison of the gain (MDC, IMI vs. IMI, BF) in the labor intensity of searching for originals, in times

| | IMI | BF |
|---|---|---|
| MDC | 3.488 | 1039.673 |
| MDC Theoretical | 16.355 | 4874.231 |
| MDC Synthetic | 24.072 | 7174.194 |
| IMI | - | 298.036 |

Table 10

Comparison of the gain (MDC, IMI vs. IMI, BF) in the labor intensity of searching for modifications, in times

| | IMI | BF |
|---|---|---|
| MDC | 0.704 | 117.292 |
| MDC Theoretical | 0.560 | 92.591 |
| MDC Synthetic | 2.905 | 483.805 |
| IMI | - | 166.565 |

For the search for originals, MDC performs significantly better than BF (more than 1 000 times) and 3 times better than IMI. The theoretical and artificially generated results are even better, with scores of about 5 000 and 7 000, respectively. However, as for the search for modifications, it is better in MDC compared to BF by about 100 times, the theoretical result is also around this value, and the artificially filled MDC result is 500 times better. However, MDC's performance is inferior to IMI's and is 0.704 of its performance. The theoretical values are even lower, but the artificially filled MDC is 2.905 times better. This is due to the mechanism of descriptor selection. MDC performs a wave search, while IMI checks the cells

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

sequentially, so MDC is faster, while IMI is more efficient in terms of the number of comparisons.

As noted above, the results of using MDC placement in the database are slightly worse than those in the RAM version. The descriptor distribution after optimization is the same as in the RAM version. The speed of optimization is 85 s, which is about 2 times faster than IMI. The search quality and labor intensity indicators are similar to the results of the RAM implementation. The search speed is about 1.5 times lower for searching for originals and 3 times lower for searching for modifications compared to IMI. However, as noted, their comparison is not very correct.

The full version of the experimental results is available on Google Drive [31].

Babenko et al. did not provide a comparison with BF search model in their paper [19], so we cannot compare the results obtained in this paper with theirs.

## 5. Conclusions

In this paper, we investigate the current problem of the low efficiency of content-based image retrieval in Big Data storage.

To solve this problem, we propose a multidimensional data model, MDC, which uses image descriptors to retrieve images and divides the descriptor feature space into subspaces, thereby performing effective image clustering. To implement MDC, several tasks were solved, such as: developing an algorithm for processing descriptors, an algorithm for optimizing the number of cells and their boundaries, methods for placing the model in computer memory, and searching and comparing descriptors.

The results of experiments comparing MDC with Inverted Multi-Index (IMI) and brute-force (BF) search approaches show that MDC has the highest speed of model construction and optimization. It is a leader in terms of search speed for both originals and image modifications. The search quality is on par with that of competitors. It is a leader in terms of labor intensity of searching for originals, but loses to IMI in this indicator for searching for originals and modifications. In this regard, the following steps to improve the implementation were identified:

– improving the algorithm for optimizing cell boundaries so that descriptors are placed more evenly, thereby reducing the search time during the execution of the search wave;

– use of a gradual search wave to reduce the labor intensity of the search;

– implementation of the parallel search algorithm.

The search algorithm and MDC structure have a good potential for parallel computing, which will improve the already good search speed.

**Contributions of authors:** conceptualization, methodology – **Serhii Smelyakov**; formulation of tasks, analysis – **Serhii Smelyakov**, **Stanislav Danylenko**; review and analysis of references, development of model, software, verification – **Stanislav Danylenko**; analysis of results – **Serhii Smelyakov**, **Stanislav Danylenko**, visualization, writing – original draft preparation – **Stanislav Danylenko**, writing – review and editing – **Serhii Smelyakov**.

### Conflict of Interest

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, authorship or otherwise, that could affect the research and its results presented in this paper.

### Financing

This study was conducted without financial support.

### Data Availability

The work has associated data in the data repository. The source code of the software will be made available upon reasonable request.

### Use of Artificial Intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

All the authors have read and agreed to the published version of this manuscript.

## References

1. Joshi, S. *35+ Google Search Statistics to Adapt to The Latest Trends*. Available at: https://learn.g2.com/google-search-statistics (accessed 30.11.2024).

2. Kumar, N. *How Many Google Searches Per Day (2024 Statistics)*. Available at: https://www.demandsage.com/google-search-statistics/ (accessed 30.11.2024).

3. Li, X., Yang J., & Ma J. Recent developments of content-based image retrieval (CBIR). *Neurocomputing*, 2021, vol. 452, pp. 675-689. DOI: 10.1016/j.neucom.2020.07.139.

4. Padma, Y. Advancements in Non-Linear Content-Based Image Retrieval (CBIR) Systems for Image Analysis. *Communications on Applied Nonlinear Analysis*, 2024, vol. 31, no. 2s, pp. 253-265. DOI: 10.52783/cana.v31.639.

5. Hirwane, R. Fundamental of Content Based Image Retrieval. *International Journal of Computer Science and Information Technologies*, 2012, no. 3, pp. 3260-3263.

6. Long, F., Zhang, H., & Feng, D. D. *Fundamentals of Content-Based Image Retrieval*. Signals and Communication Technology, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 1-26. DOI: 10.1007/978-3-662-05300-3_1.

7. Zheng, L., Yang Y., Tian, Q. SIFT Meets CNN: A Decade Survey of Instance Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018, vol. 40, no. 5, pp. 1224-1244. DOI: 10.1109/tpami.2017.2709749.

8. Li, Y., Shapiro, L. O., & Bilmes, J. A. A generative/discriminative learning algorithm for image classification. *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, IEEE, Beijing, China, 2005, vol. 2, pp. 1605-1612, DOI: 10.1109/iccv.2005.7.

9. Sivic, J., & Zisserman, A. Video Google: Efficient Visual Search of Videos. *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, vol. 4170, pp. 127-144. DOI: 10.1007/11957959_7.

10. Babenko, A. Slesarev, A. Chigorin, A., & Lempitsky, V. Neural Codes for Image Retrieval. *Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science*, Springer, Cham., 2014, vol, 8689, pp. 584-599. DOI: 10.1007/978-3-319-10590-1_38.

11. Razavian, A. S., Azizpour, H., Sullivan, J., & Carlsson, S. CNN Features off-the-shelf: an Astounding Baseline for Recognition. *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Columbus, OH, USA, 2014, pp. 512-519. DOI: 10.1109/CVPRW.2014.131.

12. Gordo, A., Almazan, J., Revaud, J., & Larlus, D. End-to-end Learning of Deep Visual Representations for Image Retrieval. *Int J Comput Vis*, 2017, vol. 124, pp. 237-254. DOI: 10.1007/s11263-017-1016-8.

13. Datar, M., Immorlica, N., Indyk, P., & Mirrokni, V. S. Locality-sensitive hashing scheme based on p-stable distributions. *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, ACM, New York, NY, USA, 2004, pp. 253-262. DOI: 10.1145/997817.997857.

14. Lin, G., Shen, C., Shi, Q., van den Hengel A., & Suter, D. Fast Supervised Hashing with Decision Trees for High-Dimensional Data. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, IEEE, 2014, pp. 1971-1978. DOI: 10.1109/cvpr.2014.253.

15. Zhang, D., Islam, M. M., Lu, G., & Hou, J. Semantic Image Retrieval Using Region Based Inverted File. *2009 Digital Image Computing: Techniques and Applications*, Melbourne, VIC, Australia, IEEE, 2009, pp. 242-249. DOI: 10.1109/dicta.2009.48.

16. Berman, A. P., & Shapiro, L. G. A flexible image database system for content-based retrieval. *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170)*, IEEE Comput. Soc., Brisbane, QLD, Australia, 1998, pp. 894-898. DOI: 10.1109/icpr.1998.711295.

17. Jégou, H., Douze, M., & Schmid, C. Product Quantization for Nearest Neighbor Search, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, 2011, vol. 33, no. 1, pp. 117-128. DOI: 10.1109/tpami.2010.57.

18. Jegou, H., Tavenard, R., Douze, M., & Amsaleg, L. Searching in one billion vectors: Re-rank with source coding. *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, IEEE, 2011, pp. 861-864. DOI: 10.1109/ICASSP.2011.5946540.

19. Babenko, A., & Lempitsky, V. The inverted multi-index. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, USA, IEEE, 2012, pp. 3069-3076. DOI: 10.1109/CVPR.2012.6248038.

20. Ge, T., He, K., Ke, Q., & Sun, J. Optimized Product Quantization for Approximate Nearest Neighbor Search. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, Portland, OR, USA, IEEE, 2013, pp. 2946-2953, DOI: 10.1109/cvpr.2013.379.

21. Mensah, M. E., Li, X., Lei, H., Obed, A., & Bombie, N. C. Improving Performance of Colour-Histogram-Based CBIR Using Bin Matching for Similarity Measure. *Artificial Intelligence and Security*, Springer International Publishing, Cham, 2020, vol. 12239, pp. 586-596. DOI: 10.1007/978-3-030-57884-8_52.

22. Guldogan, E., & Gabbouj, M. Feature selection for content-based image retrieval. *Signal, Image and Video Processing*, 2008, vol. 2, iss. 3, pp. 241-250. DOI: 10.1007/s11760-007-0049-9.

23. Guldogan, E., & Gabbouj, M. System profiles in content-based image indexing and retrieval. *Signal, Image and Video Processing*, 2009, vol. 4, iss. 4, 463-480. DOI: 10.1007/s11760-009-0137-0.

24. Qazanfari, H., AlyanNezhadi, M. M., & Khoshdaregi, Z. N. Advancements in Content-Based Image Retrieval: A Comprehensive Survey of Relevance Feedback Techniques, *arXiv.Org*, 2023. Available at: https://arxiv.org/abs/2312.10089 (accessed: 01.12.2024).

25. Muja, M., & Lowe, D. G. Fast approximate nearest neighbors with automatic algorithm configuration. *Proceedings of the Fourth International Conference on Computer Vision Theory and Applications (VISIGRAPP 2009)*, 2009, vol. 1, pp. 331-340. DOI: 10.5220/0001787803310340.

26. Qi, J. *Faiss*. GitHub. Available at: https://github.com/facebookresearch/faiss/wiki (accessed 22.12.2024).

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

27. Lux, M., & Chatzichristofis, S. A. Lire: lucene image retrieval. *Proceedings of the 16th ACM International Conference on Multimedia*, ACM, New York, NY, USA, 2008, pp. 1085-1088. DOI: 10.1145/1459359.1459577.

28. Jensen, C. S., Pedersen, T. B., & Thomsen, C. *Multidimensional Databases and Data Warehousing*, Springer Cham, 2010. 111 p. DOI: 10.2200/s00299ed1v01y201009dtm009.

29. EsotericSoftware. *GitHub - EsotericSoftware/kryo: Java binary serialization and cloning: fast, efficient, automatic*. Available at: https://github.com/EsotericSoftware/kryo (accessed 23.12.2024).

30. *COCO, Common Objects in Context*. Available at: https://cocodataset.org/ (accessed 5.12.2024)

31. Danylenko, S. *MDC-2025-ukr-art-1, Google Drive*. Available at: https://drive.google.com/drive/folders/1iTBbD1dKPxGnAAKOhXFRBDdO31jcRQf9?usp=sharing (accessed 22.12.2024).

*Received 03.01.2025, Accepted 17.02.2025*

## РОЗРОБКА БАГАТОВИМІРНОЇ МОДЕЛІ ДАНИХ ДЛЯ ЕФЕКТИВНОГО ПОШУКУ ЗОБРАЖЕНЬ НА ОСНОВІ ВМІСТУ В СХОВИЩАХ ВЕЛИКИХ ДАНИХ

***С. Д. Даниленко, С. В. Смеляков***

**Об'єктом** дослідження є пошук зображень на основі контенту. **Предметом** дослідження є моделі і методи пошуку зображень на основі контенту у сховищах великих даних в умовах високої інтенсивності надходження пошукових запитів. **Метою** дослідження є розробка багатовимірної моделі даних і пов'язаних з нею методів пошуку, яка може використовувати і адаптуватися під уже існуючі дескриптори зображень і виконувати пошук на основі них. **Завдання** полягає у: аналізі сучасних підходів і рішень для ефективного пошуку зображень на основі контенту, формулювання проблеми і вимог до системи пошуку; розробці моделі, яка буде ефективно обробляти дескриптори і розміщувати всередині таким чином, щоб мінімізувати кількість дескрипторів, з якими треба виконати порівняння під час пошуку; розробці алгоритмів пошуку; розробці метрик, виконанні експериментів і порівнянні отриманих результатів з аналогами. **Методологія** включає в себе аналіз процесу пошуку та виділення етапів формування дескриптору, його розміщення в моделі, визначення міри схожості та порівняння і формування результатів; побудова моделі і її розміщення в пам'яті; проведення експериментів з наявними в мережі Інтернет наборів даних; оцінка ефективності пошуку і формування результуючих таблиць для порівняння з аналогами. Були отримані такі **результати**: розроблена модель багатовимірного кубу (MDC) з алгоритмами оптимізації і пошуку, яка була порівняна з пошуком повним перебором та пошуком з використанням Inverted Multi-Index (IMI). Отримані результати експерименту показали, що MDC забезпечує найкращу швидкість пошуку серед конкурентів. Демонструє якість пошуку на рівні конкурентів. Трудомісткість пошуку є найкращою для пошуку оригінальних зображень у сховищі (перевірки, чи наявні вони).Трудомісткість пошуке модифікацій зображень є кращою, ніж у пошуку повним перебором більш ніж у 100 разів, однак гіршою на 30 відсотків, ніж при використанні IMI. **Висновки:** розроблена модель MDC та її алгоритм пошуку вирішує поставлену задачу ефективного пошуку зображень на основі контенту, використовуючи наявні дескриптори зображень. Отримані результати є задовільними, однак перспективним напрямком є покращення алгоритму оптимізації меж комірок та застосування паралельних обчислень.

**Ключові слова:** багатовимірна модель даних; модель пошуку; пошук зображень на основі вмісту; великі дані; обробка зображень; сховище зображень; база даних властивостей.

**Даниленко Станіслав Дмитрович** – асп. каф. програмної інженерії, Харківський національний університет радіоелектроніки, Харків, Україна.

**Смеляков Сергій Вячеславович** – д-р фіз.-мат. наук, проф., проф. каф. Програмної інженерії, Харківський національний університет радіоелектроніки, Харків, Україна.

**Stanislav Danylenko** – PhD Student, Software Engineering Department, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine,
e-mail: stanislav.danylenko@nure.ua, ORCID: 0000-0002-8142-3018, Scopus Author ID: 57816229800.

**Serhii Smelyakov** – Doctor of Mathematics, Professor at the Software Engineering Department, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine,
e-mail: serhii.smeliakov@nure.ua, ORCID: 0000-0002-5791-2479, Scopus Author ID: 24527617600.